Recorrido de Matrices

Como introducción se mencionará que es muy importante el manejo fluido de los indices para acceder a cualquier posición de la matriz. Una forma de lograr esto es haciendo ejercitación procurando que el alumno consiga seguir ciertos patrones o trayectorias de recorridos. En este sentido, el recorrido de matrices permite el acceso a los datos de un arreglo bidimensional de una forma indicada por un camino o recorrido predefinido con anterioridad.

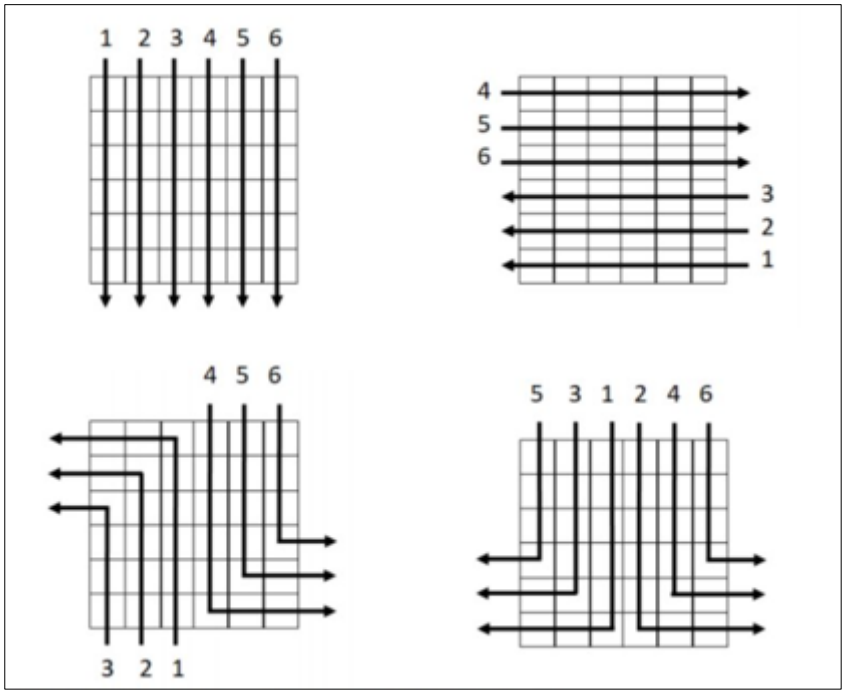


Fig. 1: Diferentes recorridos en arreglos bidimensionales.

Paso 1 (Idea): Se define la estructura de ciclos de acuerdo con la forma en que se inicia cada recorrido y la cantidad de tramos rectos que incluye cada uno.

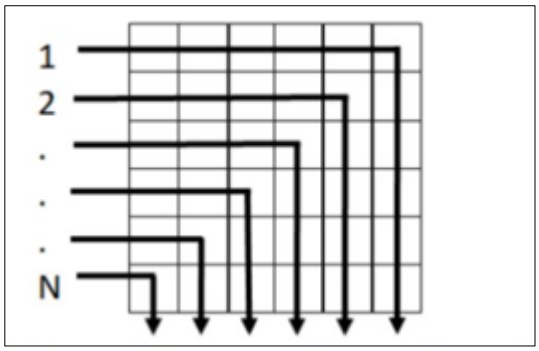


Fig. 2: Recorrido elegido.

Paso 2 (Inicio): Se observa solo la primera celda de cada recorrido, y se analiza si comparten filas o columnas en común. En este caso, todos comienzan en la primer columna, entonces decimos que la estructura es:

- Por fila (lo que varía), columna 1 fija.
- Sentido ascendente: desde la fila 1 hasta la fila N

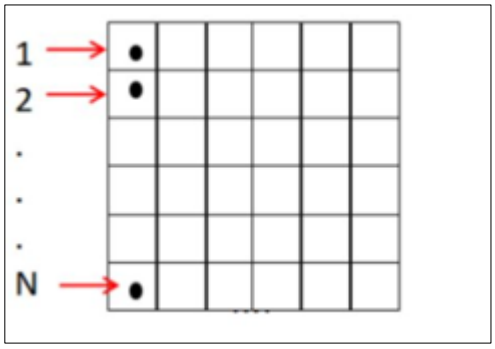


Fig. 3: Análisis de recorrido.

Paso 3:

Tramo 1 Ahora solo se observa el tramo horizontal. Se ve que siempre comienza en la primer columna y se extiende hasta una columna distinta en cada caso. En particular, se llega hasta la "Diagonal Secundaria". Entonces se dice que la estructura es:

- Por columna (lo que varía), fila fija en cada tramo.
- Sentido ascendente: desde columna 1 hasta diagonal secundaria.

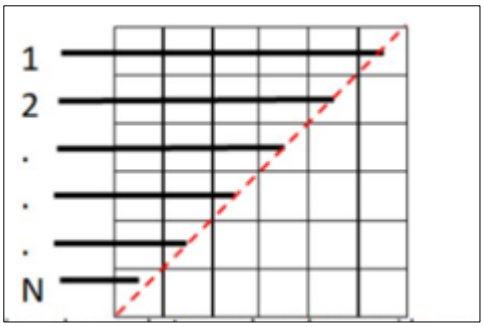


Fig. 4: Tramo 1

Tramo 2: Queda analizar el segundo tramo de cada recorrido que corresponde al movimiento vertical. Se ve que cada tramo comienza una celda por debajo de la diagonal secundaria, es decir, con una fila más, y se extiende hasta la última fila en todos los casos.
Entonces decimos que la estructura es:

- Por fila (lo que varía), columna fija en cada tramo.
- Sentido ascendente: desde diagonal secundaria hasta última fila.

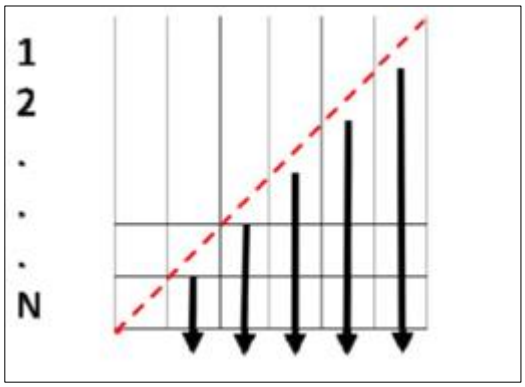


Fig. 5: Tramo 2.

Para realizar la implementación, se debe tener en cuenta que cada ciclo “Inicio” será un ciclo externo, y cada “Tramo” será un ciclo interno. En este ejemplo se tiene:

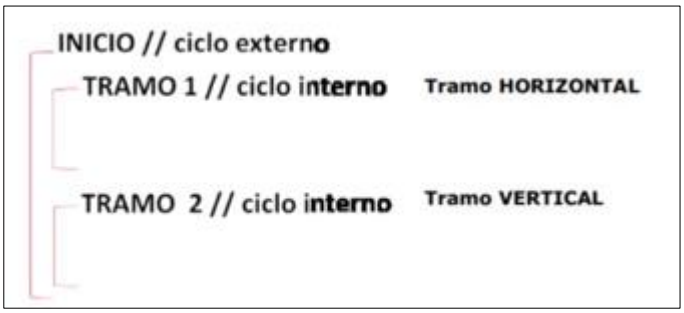


Fig. 6: Estructura general.

Ejemplo de la aplicación de la estrategia

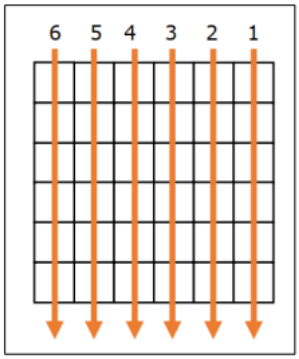
```
//Recorrido: "fila" me indica en que línea horizontal estoy
for(int fila=0; fila<n; fila++){
    /*Tramo 1 Horizontal: Me muevo hasta el nº total de celdas
    menos la fila en la que estoy, o sea diagonal secundaria
    (n-fila)*/
    for(int i=0; i<n-fila; i++){
        cout<<matriz[fila][i]<<"\t";
    }
    /*Tramo 2 Vertical: Me muevo siempre hasta el final, o sea n
    pero lo hago desde una celda por debajo de "fila" (fila +1),
    y debo ubicarme en la columna correspondiente (n-fila-1)*/
    for(int i=fila+1; i<n; i++){
        cout<<matriz[i][n-fila-1]<<"\t";
    }
    cout<<endl;
}
```

Fig. 7: Código de la estrategia en C++

Ejercicios Resueltos

Para todos los ejercicios “n” es el número de filas y columnas, ya que los ejercicios son matrices cuadradas, donde “n” es par. También, es necesario aclarar que los índices mayores de las matrices al igual que los arreglos van desde 0 hasta (n – 1).

Ejercicio A



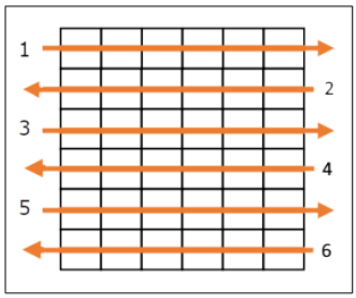
Inicio: varía columnas en forma descendente, desde n-1 hasta 0

Tramo único (vertical):
Varía filas en forma ascendente, desde 0 hasta n-1

Código:

```
//Recorrido
for(int col=n-1; col>-1;col--){
    for(int fila=0; fila<n; fila++){
        cout<<matriz[fila][col]<<"\t";
    }
    cout<<endl;
}
```

Ejercicio B



Inicio: varía filas en forma ascendente, desde 0 hasta n-1

Dependiendo si es par o impar el número de fila, tenemos que hacer un recorrido hacia la derecha o hacia la izquierda

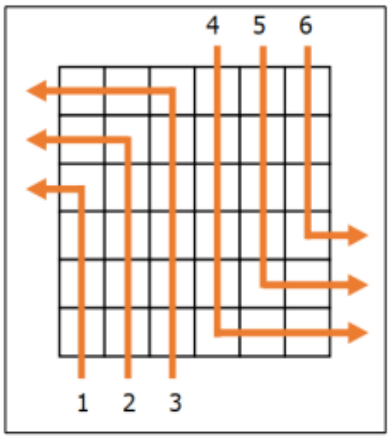
Tramo 1 (horizontal de Izq a Der): Varía columnas en forma ascendente, desde 0 hasta n-1

Tramo 2 (horizontal de Der a Izq): Varía columnas en forma descendente, desde n-1 hasta 0

Código:

```
//Recorrido
for(int fila=0; fila<n;fila++){
    if (fila%2 == 0){
        //Tramo 1
        for(int col=0; col<n; col++){
            cout<<matriz[fila][col]<<"\t";
        }
    }
    else{
        //Tramo 2
        for(int col=n-1; col>-1; col--){
            cout<<matriz[fila][col]<<"\t";
        }
    }
    cout<<endl;
}
```

Ejercicio C: 3 opciones diferentes



<u>Opción 1</u>: Un solo inicio y cuatro tramos

Inicio: Varía columnas en forma ascendente, desde 0 hasta n-1

Dependiendo si estamos en la mitad izquierda de la matriz, vamos a recorrer de una forma y si estamos en la mitad derecha de la matriz, vamos a recorrer de otra forma diferente.

Lado izquierdo de la matriz:

Tramo 1 (vertical de abajo hacia arriba): Varía filas en forma descendente, desde n-1 hasta un valor distinto en cada en cada paso, este valor desciende de a 1.

Tramo 2 (horizontal de derecha a izquierda): Varía columnas en forma descendente, desde un valor distinto en cada paso hasta 0, este valor asciende de a 1.

Lado derecho de la matriz:

Tramo 3 (vertical de arriba hacia abajo): Varía filas en forma ascendente, desde 0 hasta un valor distinto en cada paso, este valor desciende de a 1.

Tramo 4 (horizontal de izquierda a derecha): Varía columnas en forma ascendente, desde un valor distinto en cada paso hasta n-1, el valor asciende de a 1.

Código:

```
//Recorrido
cout<<"Comienzo de Recorrido...\n";
//Inicio
for(int col_in=0; col_in<n; col_in++){
    if(col_in <(n/2)){
        //Tramo 1
        for(int fila=n-1; fila>=(n/2)-1-col_in; fila--){
            cout<<matriz[fila][col_in]<<"\t";
        }
        //Tramo 2
        for(int col=col_in-1;col>-1; col--){
            cout<<matriz[(n/2)-1-col_in][col]<<"\t";
        }
        cout<<endl;
    }else{
        //Tramo 3
        for(int fila=0; fila<=n+((n/2)-1)-col_in; fila++){
            cout<<matriz[fila][col_in]<<"\t";
        }
        //Tramo 4
        for(int col= col_in+1; col<n; col++){
            cout<<matriz[n+((n/2)-1)-col_in][col]<<"\t";
        }
        cout<<endl;
    }
    cout<<endl;
}
```

Opción 2: Dos inicios y dos tramos para cada uno de los inicios. Cada uno de estos inicios representa una mitad de la matriz.

Inicio 1: Varía columnas en forma ascendente, desde 0 hasta ((n/2)-1).

Tramo 1 (vertical de abajo hacia arriba): Varía filas en forma descendente, desde n-1 hasta un valor distinto en cada en cada paso, este valor desciende de a 1.

Tramo 2 (horizontal de derecha a izquierda): Varía columnas en forma descendente, desde un valor distinto en cada paso hasta 0, este valor asciende de a 1.

Inicio 2: Varía columnas de forma ascendente, desde (n/2) hasta n-1.

Tramo 3 (vertical de arriba hacia abajo): Varía filas en forma ascendente, desde 0 hasta un valor distinto en cada paso, este valor desciende de a 1.

Tramo 4 (horizontal de izquierda a derecha): Varía columnas en forma ascendente, desde un valor distinto en cada paso hasta n-1, el valor asciende de a 1.

Código:

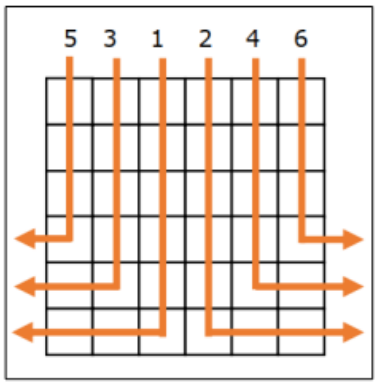
```
//Inicio 1
for(int col=0; col<(n/2); col++){
   //Tramo 1
    for(int i=n-1; i>=(n/2)- col; i--){
        cout<<matriz[i][col]<<"\t";
    }
   //Tramo 2
    for(int i=col; i>-1; i--){
        cout<<matriz[(n/2)-1-col][i]<<"\t";
    }
    cout<<endl;
}

//Inicio 2
for(int col=(n/2); col<n; col++){
    //Tramo 1
    for(int i=0; i< n+(n/2)-col-1;i++){
        cout<<matriz[i][col]<<"\t";
    }
   ///Tramo 2
    for(int i=col; i<n; i++){
        cout<<matriz[n+(n/2)-col-1][i]<<"\t";
    }
    cout<<endl;
}
```

<u>Opción 3:</u> Dos inicios y dos tramos para cada uno de los inicios. Cada uno de estos inicios representa una mitad de la matriz. Y colocamos 2 variables que van a manejar los valores variables de filas y columnas. La estrategia es similar a la anterior, solo que modificamos las variables topes de fila y columna.

Código:

```
//Recorrido
cout<<"Comienzo de Recorrido...\n";
int cont_col=-1; //Inicio de contador columnas
int cont_fila=2; //Inicio de contador filas. Equivalente ((n/2)-1)
//Inicio
for(int col_in = 0; col_in < n; col_in++){
    if(col_in<n/2){
        //Tramo 1
        for(int fila = n-1; fila>=cont_fila; fila--){
            cout<<matriz[fila][col_in]<<"\t";
        }
        //Tramo 2
        for(int col= cont_col; col>-1; col--){
            cout<<matriz[cont_fila][col]<<"\t";
        }
        cout<<endl;
        if(col_in==(n/2)-1){
            cont_fila=n-1;      //Reseteo el contador fila
            cont_col=(n/2)+1;   //Reseteo el contador columna
        }else{
            cont_fila--;        //Decremento el contador fila
            cont_col++;         //Incremento el contador columna
        }
    }else{
        //Tramo 3
        for(int fila = 0; fila<=cont_fila; fila++){
            cout<<matriz[fila][col_in]<<"\t";
        }
        //Tramo 4
        for(int col= cont_col; col<n; col++){
            cout<<matriz[cont_fila][col]<<"\t";
        }
        cout<<endl;
        cont_fila--;        //Decremento el contador fila
        cont_col++;         //Incremento el contador columna
    }
}
```

Ejercicio D



Vamos a usar un inicio que se mueva solo en una mitad de la matriz y en cada paso vamos a imprimir dos recorridos, cada uno de estos tendrá dos tramos.

Inicio (mitad izquierda de la matriz): Varía columnas en forma descendente, desde ((n/2)-1) hasta 0. En cada paso imprimimos dos recorridos:

- primer paso recorrido 1 y 2
- segundo paso recorrido 3 y 4
- tercer paso recorrido 5 y 6

Empieza primer recorrido de cada paso:

Tramo 1 (vertical de abajo hacia arriba): Varía filas en forma ascendente, desde 0 hasta un valor distinto en cada paso, este valor desciende de a 1.

Tramo 2 (horizontal de derecha a izquierda): Varía columnas en forma descendente, desde un valor distinto en cada paso hasta 0, este valor desciende de a 1.

Empieza segundo recorrido de cada paso_

Tramo 3 (vertical de abajo hacia arriba): Varía filas en forma ascendente, desde 0 hasta un valor distinto en cada paso, este valor desciende de a 1.

Tramo 4 (horizontal de izquierda a derecha): varía columnas en forma ascendente, desde un valor distinto en cada paso hasta n-1, este valor asciende de a 1.

Código:

```
//Recorrido
cout<<endl<<"Comienza recorrido \n"<<endl;
//Inicio
for(int col_in=(n/2)-1; col_in>-1; col_in--){
    //Tramo 1
    for(int fila=0; fila<=col_in +(n/2);fila++){
        cout<<matriz[fila][col_in]<<"\t";
    }
    //Tramo 2
    for(int col= col_in-1; col>-1; col--){
        cout<<matriz[col_in +(n/2)][col]<<"\t";
    }
    //Finaliza 1er recorrido que muestra cada paso
    cout<<endl;
    //Tramo 3
    for(int fila=0; fila<=col_in +(n/2);fila++){
        cout<<matriz[fila][n-col_in-1]<<"\t";
    }
    //Tramo 4
    for(int col=n-col_in; col<n; col++){
        cout<<matriz[col_in +(n/2)][col]<<"\t";
    }
    //Finaliza 2do recorrido que muestra cada paso
    cout<<endl;
}
```